Recent Developments to Improve Scalability of Sparse Direct Solver

X. Sherry Li

Lawrence Berkeley National Laboratory

PARA'06 Workshop

June 19, 2006

Introduction

- Sparse direct solvers are robust and reliable, but not tera/peta-scale friendly. Why?
  - Irregular, indirect memory access
  - Computational dependency
  - High communication-to-computation ratio (latency-bound)
  - Architectural trend: wider gap between processor and interconnect speeds

NEW DEVELOPMENTS:

- Switch-to-dense
  - Reduce indirect addressing and communication
- Parallel symbolic factorization [Grigori, Demmel, L.]
  - Improve memory scalability
- Optimal complexity sparse factorization [Gu, Xia, L.]

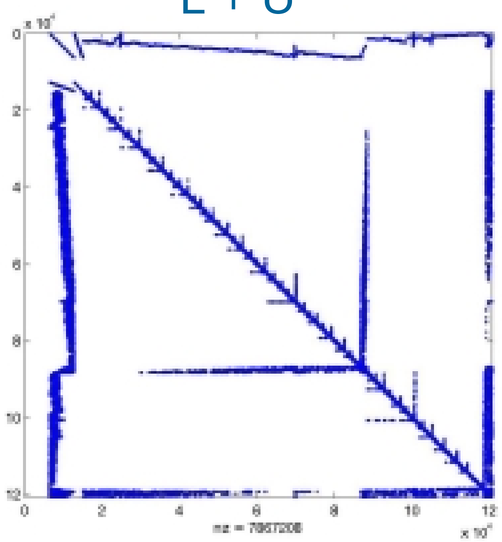
SuperLU_DIST major steps

- Static numerical pivoting: improve diagonal dominance
  - Currently use MC64; Parallelization underway [J. Riedy]
- Sparsity-preserving ordering
  - Can use parallel Metis (ParMetis)
- Symbolic factorization:
  - Being parallelized (this talk)
- Numerics: factorization, triangular solves, iterative refinement
  - Parallelized; More performance tuning, scaling

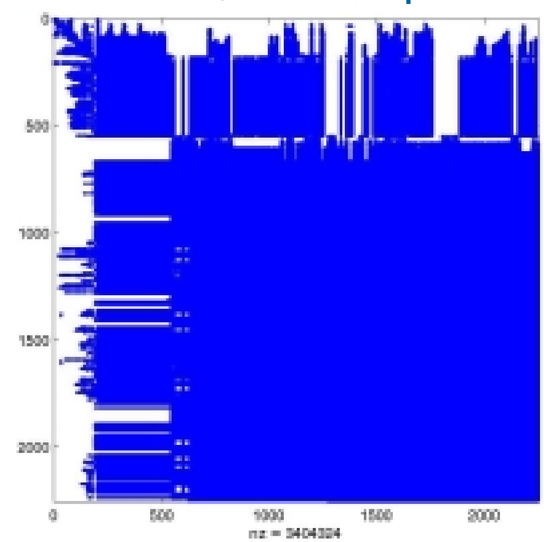
Switch-to-dense

- Factors become denser and denser towards end …
- Example: twotone (circuit), n = 120,750
- Last dense block:
  - size = 2250, density = 67%, flops >= 70%

L + U



L + U, bottom part



Switch-to-dense benefit

- IBM p575 Power5; 7.6 Gflops peak
- Use 32 processors

Matrix	*n*	*Dense size*	*Density*	*Dense Flops*	*Mflops/P*	*Time*
Twotone	120,750	2905	56%	84%	3,020	1.80
old					81	2.91
Pre2	659,033	9269	90%	83%	4,557	12.41
old					1,004	16.26
Torso3	259,156	10,444	99%	19%	4,570	41.89
old					2,862	43.41

Content

- Switch-to-dense
  - Reduce indirect addressing and communication
- Parallel symbolic factorization [Grigori, Demmel, L.]
  - Improve memory scalability
- Optimal complexity sparse factorization [Gu, Xia, L.]

Symbolic factorization

- Identify nonzero structure of L, U factors.
- Complexity: greater than nnz(L+U), but much smaller than flops(LU); Very fast in practice.
- Why parallel?
  - Matrix A may not fit in one processor.
- Why difficult?
  - Sequentiality: computation of i-th column/row depends on results of the previous ccolumns/rows.
  - Lower computation-to-communication ratio.

Parallelizing symbolic factorization

Goal:

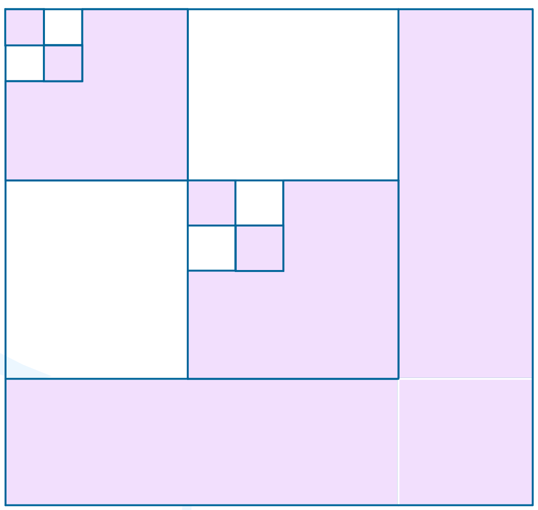
- Improve memory scalability, while maintaining reasonable speedup.

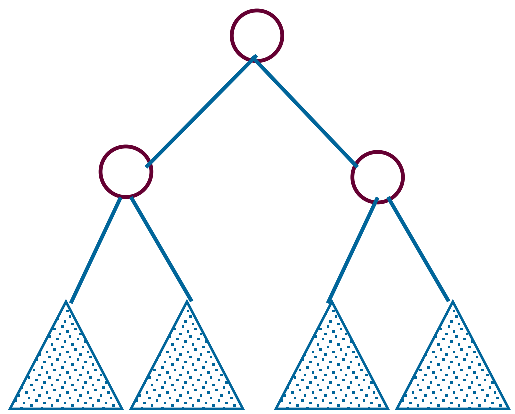
Approach:

- Use graph partitioning to reorder/partition matrix.
  - ParMetis on structure of A + A'
- Exploit parallelism given by this partition (coarse level) and by a block cyclic distribution (fine level).
- Identify dense separators, dense columns of L and rows of U to decrease computation.

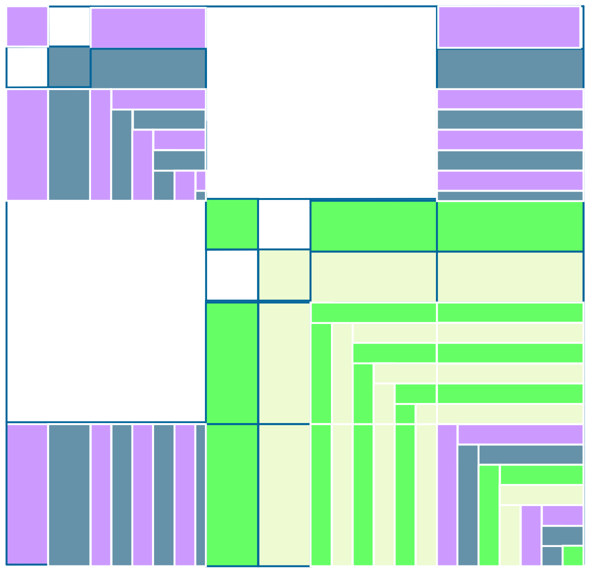
Matrix partition

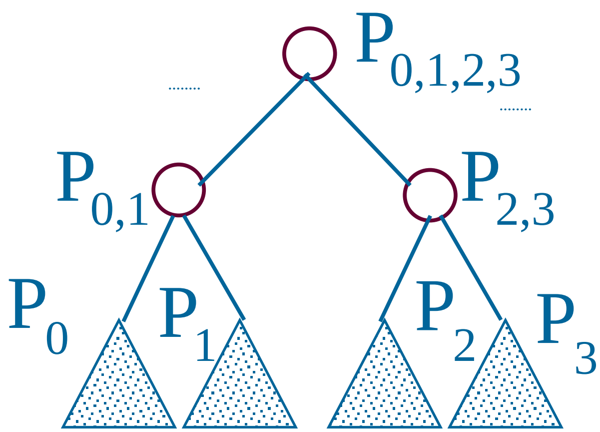
- Separator tree
  - Balanced tree with balanced data distribution
  - Exhibits computational dependencies
    - If node j updates node k, then j belongs to subtree rooted at k.





Matrix distribution





Algorithm

1. Assign all the processors to the root.
2. Distribute the root (1D block cyclic along the diagonal) to processors in the set.
3. Assign to each subtree half of the processors.
4. Go to Step 1 for each subtree which is assigned more than one processor.

Algorithm

1) Perform local symbolic factorization of leaf node

2) for each level from 1 to logP **do**

Let N(x:y) be node owned by myPE

/* inter-level computation */ — left looking

Send / Receive necessary $L(:,1:x-1),\ U(1:x-1,:)$

Use received data to update $L(:,x:y),\ U(x:y,:)$

/* intra-level computation */ — right looking

for each block (i:j) of node N

If myPE owns this block

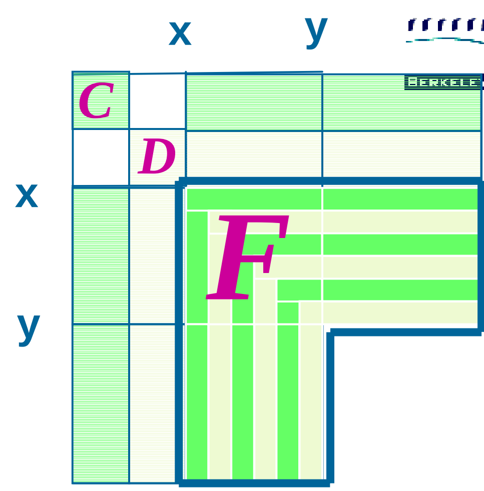
Compute $L(:,i:j), U(i:j,:)$

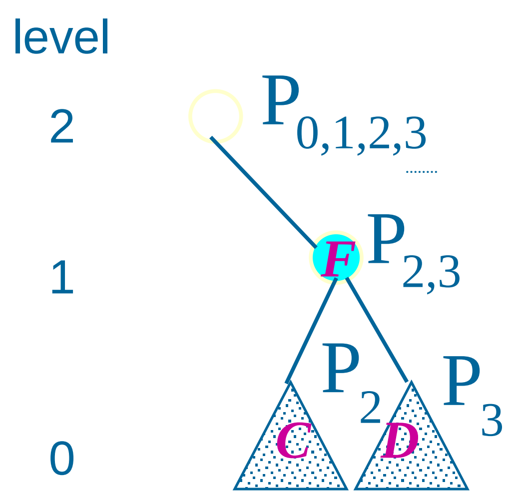
Send / Rceive block (i:j) if necessary — left looking

Use received data to update $L(:,j+1:y), U(j+1:y,:)$

endfor

endfor — right looking





Experiments

Goals:

- Compare with sequential symbolic factorization algorithm in SuperLU_DIST (SFseq).
- Analyze memory usage and parallel runtime.

Test Matrices:

- 3D regular grid model problems
- Unsymmetric matrices: circuit simulation, fluid flow

Machine:

- IBM Power3, RS/6000

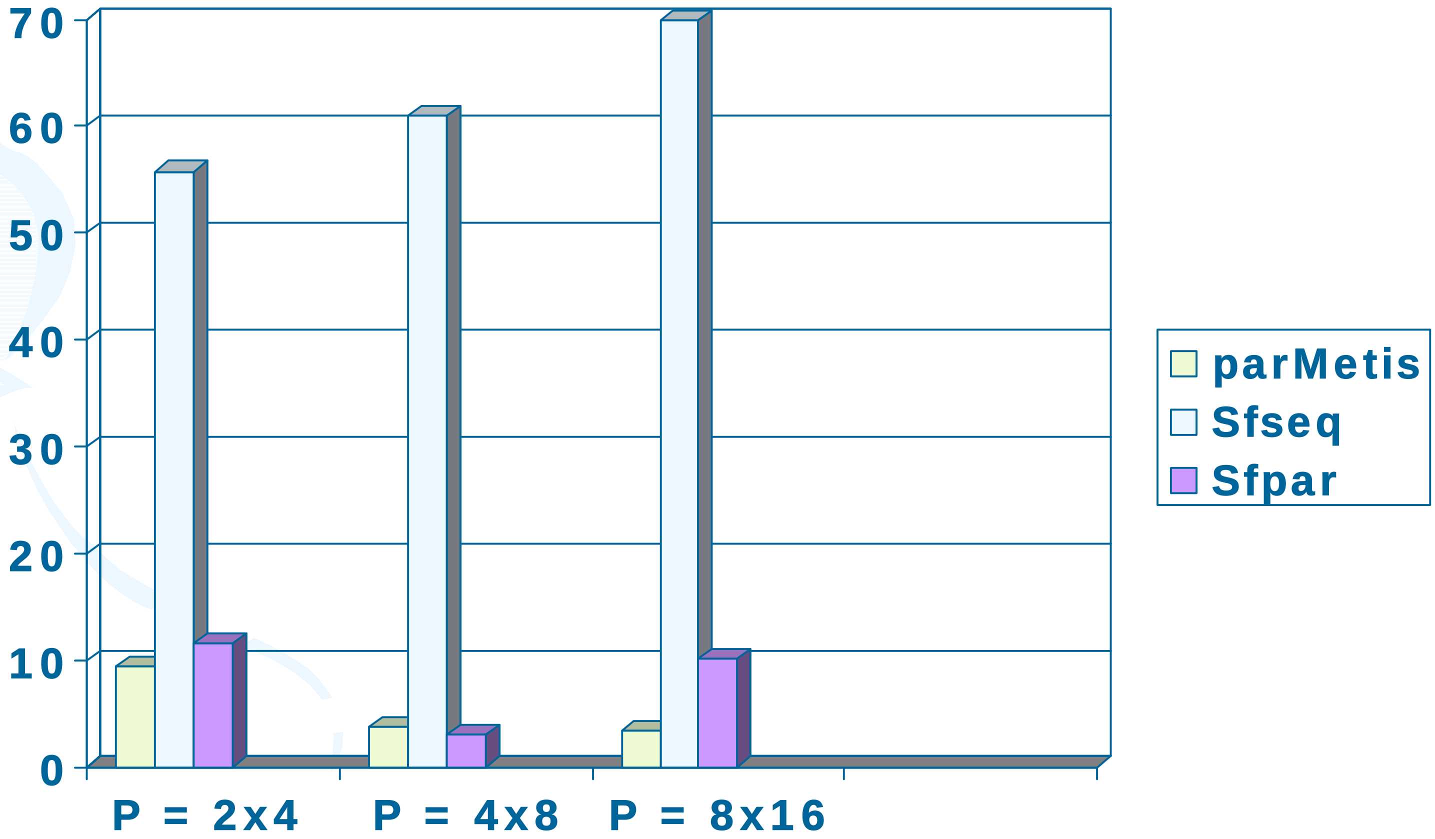
3D regular grid (1/2)

- Laplacian, cubic grid of size 90, nnz = 7.9M
- Memory usage:
  - SFseq (symbolic sequential), SFpar (symbolic parallel)
  - Entire solvers: SLU_SFseq, SLU_SFpar

Memory needs(MB)	*P=8*	*P=32*	*P=128*
Nnz(L+U)*10^6	1408.4	1498.1	1588.4
SFseq	452.1	461.9	541.4
SFpar (max)	44.9	16.7	14.2
SFseq / SFpar	10.1	27.6	38.1
Factor (max)	1540.0	403.6	108.0
SLU_SFseq	2081.8	941.1	673.3
SLU_SFpar	1723.4	521.5	187.9

3D regular grid (2/2)

- Runtime in seconds



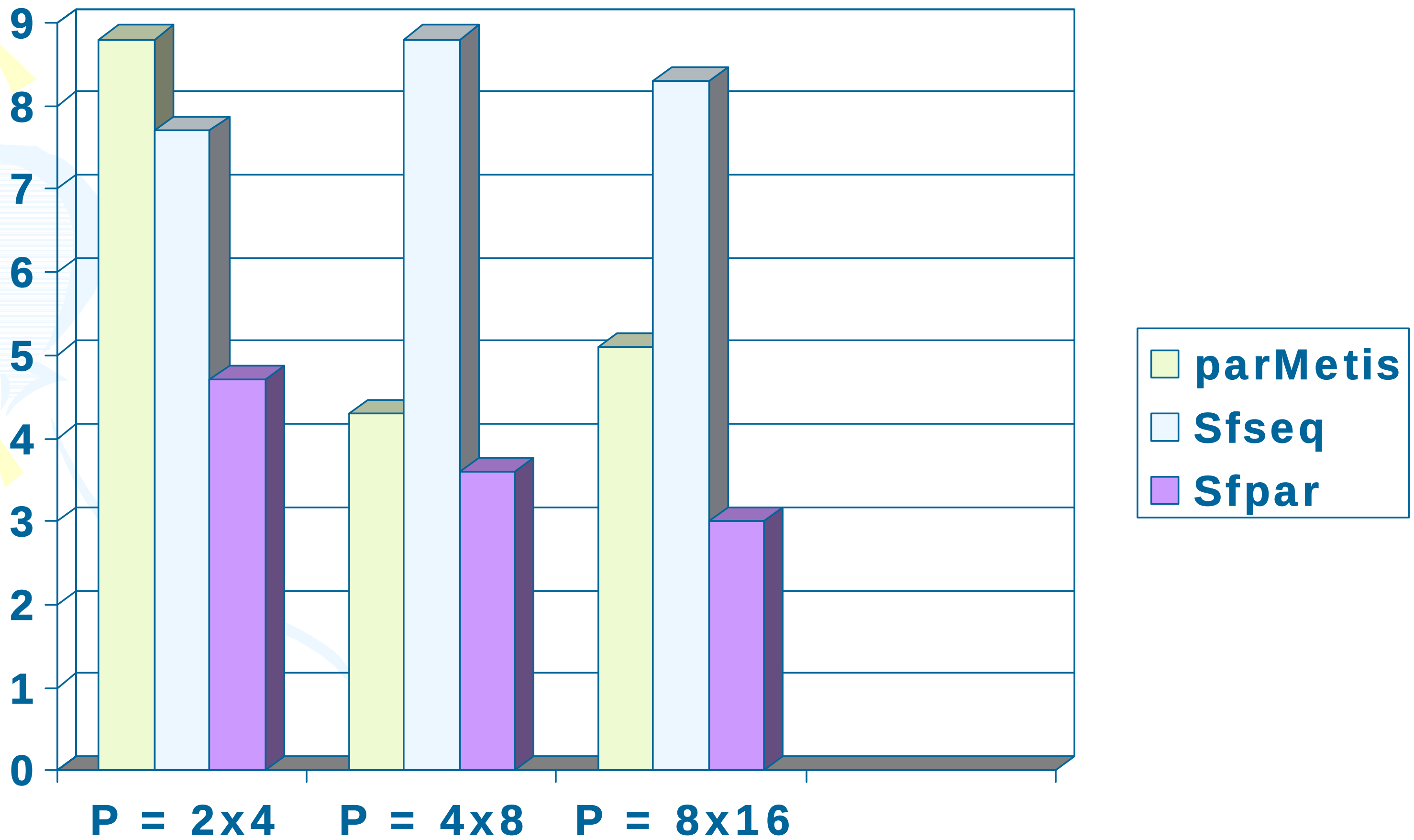
Circuit simulation (1/1)

- Pre2: n = 659,033, nnz = 5.9M, 92M fill-ins using parMetis on one processor
- Memory usage:
  - SFseq (symbolic sequential), SFpar (symbolic parallel)
  - Entire solvers: SLU_SFseq, SLU_SFpar

Memory needs(MB)	*P=8*	*P=32*	*P=128*
Nnz(L+U)*10^6	120.1	145.7	138.6
SFseq	122.0	133.0	126.4
SFpar (max)	31.5	11.0	7.2
SFseq / SFpar	3.9	12.1	17.6
Factor (max)	167.6	52.5	14.2
SLU_SFseq	415.3	239.7	159.0
SLU_SFpar	347.9	157.6	96.5

Circuit simulation (2/2)

- Runtime in seconds



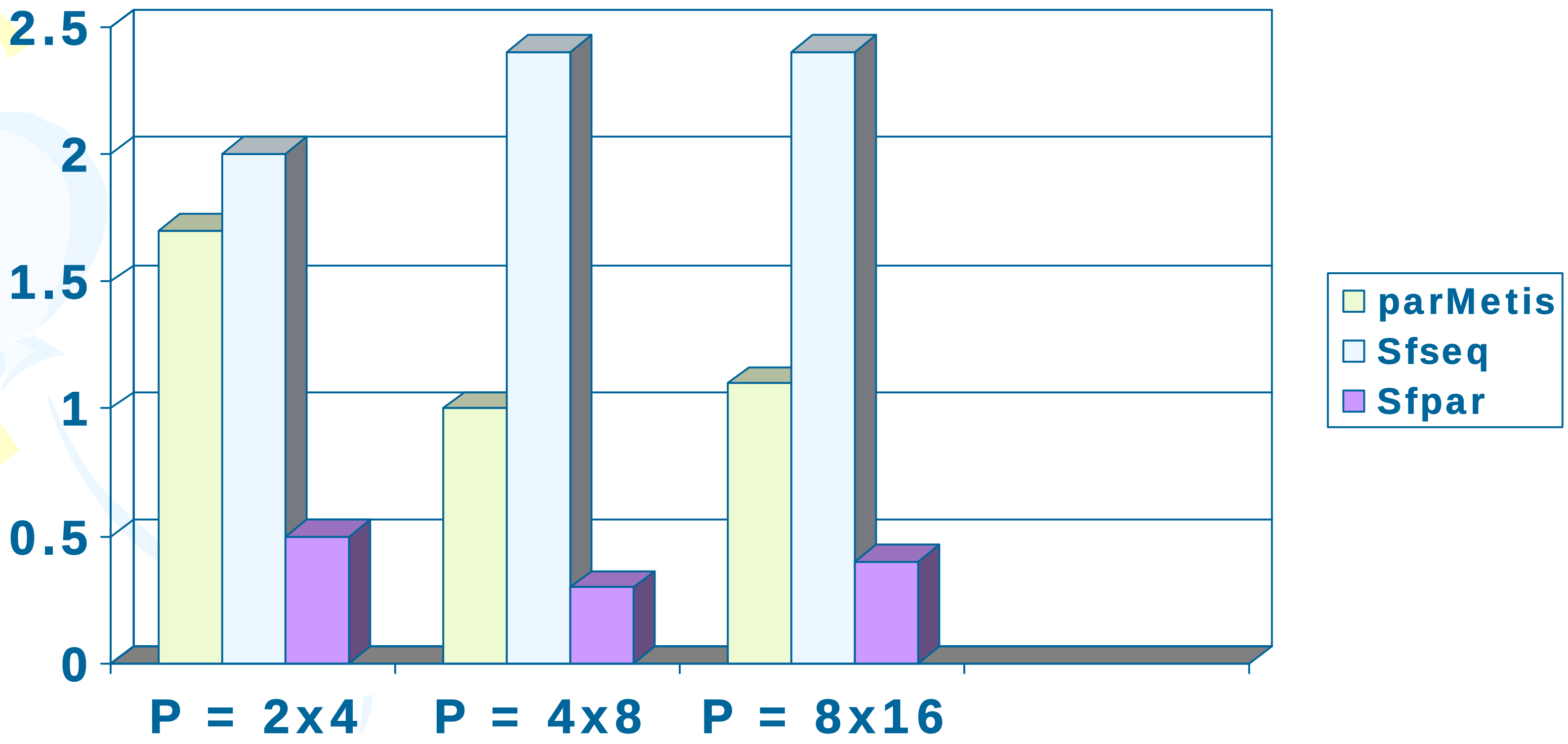
Fluid flow (1/1)

- bbmat: n = 38,744, nnz = 1.8M, 34M fill-ins using ParMetis on one processor
- Memory usage:
  - SFseq (symbolic sequential), SFpar (symbolic parallel)
  - Entire solvers: SLU_SFseq, SLU_SFpar

Memory needs(MB)	*P=8*	*P=32*	*P=128*
Nnz(L+U)*10^6	35.0	36.7	36.6
SFseq	35.6	36.5	40.7
SFpar (max)	6.7	3.0	1.6
SFseq / SFpar	5.3	12.2	25.4
Factor	44.7	13.1	4.0
SLU_SFseq	86.4	52.1	45.3
SLU_SFpar	58.4	19.5	8.0

Fluid flow (2/2)

- Runtime in seconds

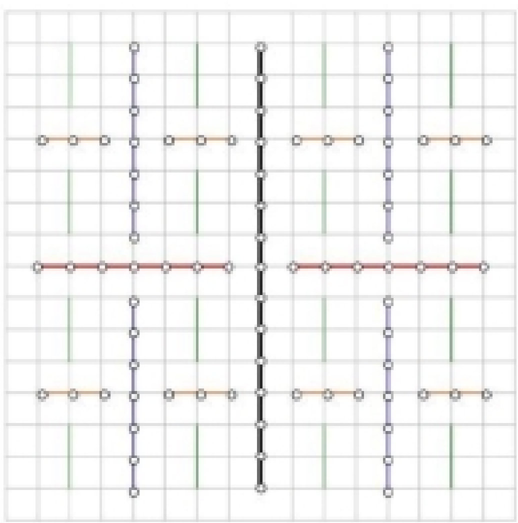


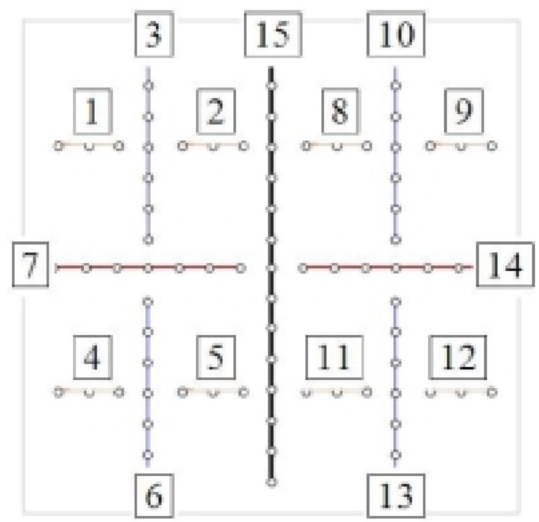
Content

- Switch-to-dense
  - Reduce indirect addressing and communication
- Parallel symbolic factorization [Grigori, Demmel, L.]
  - Improve memory scalability
- Optimal complexity sparse factorization [Gu, Xia, L.]

Fast solver

- In the spirit of fast multipole, but for matrix inversion
- Model problem: discretized system Ax = b from certain PDEs, e.g., 5-point stencil on k x k grid, n = k^2
- Nested dissection ordering gave optimal complexity in exact arithmetic [Hoffman/Martin/Ross]
  - Factorization cost: O(k^3)





Exploit low-rank property

- Consider top-level dissection:
- S is full
  - Needs O(k^3) to find u3

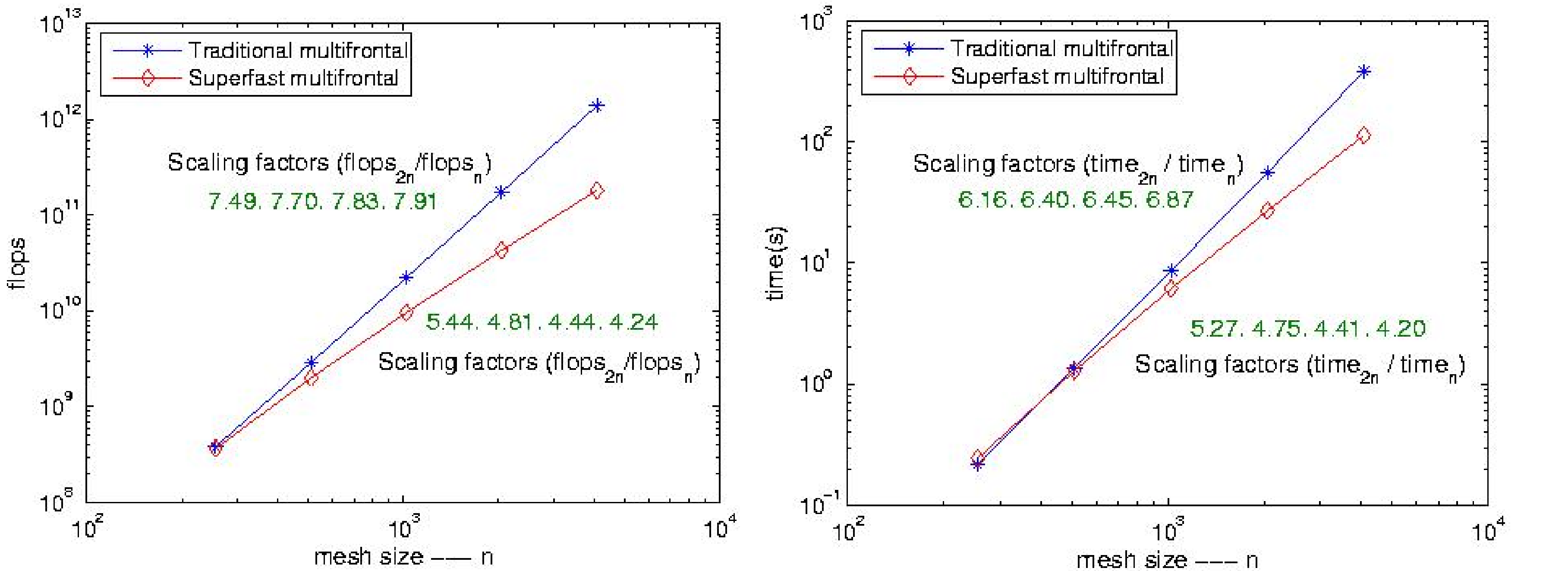
$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

$$S\, u_3 = f_3 - A_{31} A_{11}^{-1} f_1 - A_{32} A_{22}^{-1} f_2$$

- But, off-diagonal blocks of S has low numerical ranks (e.g. 10~15)
  - U_3 can be computed in O(k) flops
- Generalize to multilevel dissection: all diagonal blocks corresp. to the separators have the similar low rank structure
- Low rank structures can be represented by hierarchical semi-separable (HSS) matrices [Gu et al.] (… think about SVD)
- Factorization complexity … essentially linear
  - 2D: O(p k^2), p is related to the problem and tolerance (numerical rank)
  - 3D: O(c(p) k^3), c(p) is a polynomial of p

Results of the model problem

- Flops and times comparison



Research issues

- Analysis of 3D problems, and complex geometry
- Larger tolerance → preconditioner (another type of ILU)
  - If SPD, want all the low rank structures to remain SPD
- Performance tuning for many small dense matrices (e.g. size 10~20)
- Need a hybrid solver; find a good switching level
  - Benefits show up only for large enough mesh
- Local ordering of unknowns
  - Node ordering within a separator affects numerical ranks
- Parallelization

Summary of results

- Switch-to-dense
  - Worthwhile if dense flops consistutes over 50%
  - Up to 60% faster
- Parallel symbolic factorization
  - Memory: up to 25x reduction of symbolic fact.; up to 5x reduction of the entire solver
  - Time: up to 14x speedup of symbolic fact.; up to 20% faster of the entire solver
- Optimal complexity factorization
  - Showed linear scaling

Questions?